Sample Final Exam Solutions

BA810

Total 30 points.

Allocated time: 2h 30mins.

This is a closed book/notes/laptop exam. You may refer to the one page of notes (both sided is fine) that you prepared and brought to the exam. Use of any other reference or device is not allowed. Communication with anyone else during the exam is strictly prohibited.

---

1. (3 points) Suppose you have a dataset from a data generating process following $y = 1 + x + 2x^2$. If you fit two linear regression models to it: one with only terms linear in x and another with terms that are polynomials of x up to degree 3.
    a. Which one is more likely to overfit?
    Degree 3 polynomial
    b. Which one is likely to have high variance?
    Degree 3 polynomial
    c. Which one is likely to have high bias?
    Linear in x.
2. (1 points) Which of the following is an appropriate model to predict house price?
    a. Logistic regression
    b. Decision tree regression (✓)
3. (1 points) Which of the following strategies ensures that a predictive model is tested on every one of the data records? Choose the correct answer.
    a. Only k-fold cross validation
    b. Only leave-one-out cross validation
    c. Both a and b (✓)
    d. Neither of the two
4. (1+2 points) Which of the two regularization approaches can be used for feature selection?
    a. Lasso (✓)
    b. Ridge

    Briefly support your answer in 2-3 sentences.

Though both Ridge and Lasso shrink the coefficients towards zero, only lasso turns some of them to exactly to zero, for sufficiently high values of regularization parameters. With ridge regression, all coefficients are shrunk, but not made zero. For sufficiently high value of regularization parameter, ridge will turn all, not a subset, to zero, thus can't be used for selective a subset of features.

5. (3 points) Provide three important differences between cross validation and bootstrap (no more than 3 sentences for each difference).

Cross validation	Bootstrap
1. Used to estimate generalization error of a predictive model. 2. Relies on random partitioning of the original dataset. 3. Model is trained on a dataset of smaller size than the original.	1. Used to estimate uncertainty around estimates of interest. 2. Relies on random sampling, with replacement, of the original dataset. 3. Model is trained on a dataset of equal size as the original (might include duplicates).

6. (3 points) Consider the following confusion matrix:

Actual ↓ Predicted →	False	True
False	20	2
True	1	5

i. Compute the precision and recall for positive class.

Precision: 5/7, Recall: 5/6

ii. What is the accuracy?

Accuracy: 25/28

7. (2 points) What is the shape of the ROC curve of a random classifier? Briefly justify, in up to five sentences, why the curve must have such a shape.

The ROC curve is a plot of fraction of positives one has retrieved as positive against the fraction of negatives retrieved. The ROC curve of a random classifier is a diagonal line from lower left to upper right corner. This is so because a random classifier, one that randomly picks a record and presents it as positive, would retrieve fractions of the positives and negatives at the same

rate. For example, if there were 10 positives and 30 negatives, if we randomly select records from it, we'll retrieve one positive for every three negatives, because of the frequency at which they are present in the dataset. Since, we are plotting fractions of positives and negatives that we have retrieved, and they are equal, the curve will trace a diagonal line.

8. (2 points) Let's say you have a dataset with 10 different features of which 3 are strong predictors of an outcome of interest, but *only if* all 3 are present in the model. To carry out feature selection, would you use forward selection or backward selection? Why? Justify your answer in no more than 3 sentences.
   We should use backward selection because it'll evaluate each feature in the presence of all other features and recognize the contribution of the three complementary features. On the other hand, when forward selection evaluates the improvement to prediction brought forward by each of the three such features, without any other feature, it'll underestimate their contribution. The features don't predict well by themselves.
9. (1 points) Support vector machines require a margin between the two classes that is never violated.
   a. True
   b. False (✓)

      Note: support vector machines allow the margin to be violated at a cost.
10. (1 points) Which of the following does not require missing values to be removed or imputed before it can be applied?
    a. Logistic regression
    b. Decision tree regression (✓)
    c. Support vector machine
11. (2 points) Consider the following steps to train a model and estimate its generalization error.

We use a ridge regression, whose objective function is $RSS + \lambda \sum_j^p \beta_j^2$, where λ is the regularization parameter, β_j is the j'th coefficient, and p is the number of coefficients/covariates. We follow the following steps:

i. Establish a set of λ values between 0 and 10 at 0.1 intervals.
ii. Choose the value of λ that leads to smallest cross validation error.
iii. Re-estimate the model with the chosen value of λ using the entire dataset.
iv. Present the model obtained in step iii as the best model to use with smallest error seen in step ii as it's estimated generalization error.

What is the problem with this approach? How would you rectify it? Answer each using up to 3 sentences.

Since we used cross validation on step ii, the resulting cross validation error will likely underestimate the generalization error. One way to fix this problem is to first divide the original dataset into a training and test data. Then carry out i — iii within training data, then modify the step iv to measure the error on the test data to estimate the generalization error of the model.

12. (8 points) Write the code to learn a predictive model in python by doing the following steps:
    a. Load and summarize College.csv dataset that is present in the local directory.
    b. Prepare a pipeline containing imputation, normalization, one-hot-encoding, forward search feature selection, and a linear regression model to predict the number of Apps a college will receive.
    c. Perform a grid search to find the best number of features to use and the corresponding predictive model.

See the next few pages…

(Note, in the provided code, some snippets are for examining the data; you don't need to provide them while writing down the code. Just the code to achieve the above three steps will suffice for full point).

We'll start by loading and taking a quick look at the dataset.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from google.colab import drive

drive.mount('/content/drive')
data_folder = '/content/drive/MyDrive/Colab Notebooks/BA810/Data/'
```

```
Mounted at /content/drive
```

```
rawdata = pd.read_csv(data_folder+'College.csv', index_col=0)
rawdata.index.name = 'College'  # The first column has college names, we'll use t
rawdata.head()
```

College	Private	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Unde
Abilene Christian University	Yes	1660	1232	721	23	52	2885	
Adelphi University	Yes	2186	1924	512	16	29	2683	
Adrian College	Yes	1428	1097	336	22	50	1036	
Agnes Scott College	Yes	417	349	137	60	89	510	
Alaska Pacific University	Yes	193	146	55	16	44	249	

```
rawdata.info()
rawdata.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 777 entries, Abilene Christian University to York College of Pennsylvania
Data columns (total 18 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Private      777 non-null    object 
 1   Apps         777 non-null    int64  
 2   Accept       777 non-null    int64  
 3   Enroll       777 non-null    int64  
 4   Top10perc    777 non-null    int64  
 5   Top25perc    777 non-null    int64  
 6   F.Undergrad  777 non-null    int64  
 7   P.Undergrad  777 non-null    int64  
 8   Outstate     777 non-null    int64  
 9   Room.Board   777 non-null    int64  
 10  Books        777 non-null    int64  
 11  Personal     777 non-null    int64  
 12  PhD          777 non-null    int64  
 13  Terminal     777 non-null    int64  
 14  S.F.Ratio    777 non-null    float64
 15  perc.alumni  777 non-null    int64  
 16  Expend       777 non-null    int64  
 17  Grad.Rate    777 non-null    int64  
dtypes: float64(1), int64(16), object(1)
memory usage: 131.5+ KB
```

	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.U
count	777.000000	777.000000	777.000000	777.000000	777.000000	777.000000	7
mean	3001.638353	2018.804376	779.972973	27.558559	55.796654	3699.907336	8
std	3870.201484	2451.113971	929.176190	17.640364	19.804778	4850.420531	15
min	81.000000	72.000000	35.000000	1.000000	9.000000	139.000000	
25%	776.000000	604.000000	242.000000	15.000000	41.000000	992.000000	
50%	1558.000000	1110.000000	434.000000	23.000000	54.000000	1707.000000	3
75%	3624.000000	2424.000000	902.000000	35.000000	69.000000	4005.000000	9

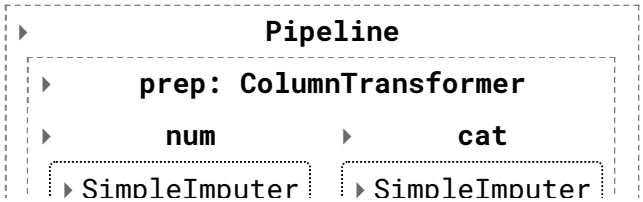
From these, we see that there is only one categorical feature — whether the college is private or not. There are no null values — all features have 777 non-null values, same as the number of rows in the dataset. Before we train and tune the model, let's split the data into training and testing so that we can estimate generalization error later.

```
1 from sklearn.model_selection import train_test_split
2 X = rawdata.drop('Apps', axis=1) # separate X ...
3 y = rawdata['Apps'].copy()       # from y
4 train_X, test_X, train_y, test_y = train_test_split(X, y, test_size = .2, random_
5 train_X.shape, test_X.shape, train_y.shape, test_y.shape
```

```
((621, 17), (156, 17), (621,), (156,))
```

We'll set up the full pipeline here, with imputation, scaling, one hot encoding, and linear regression.

```
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.feature_selection import SequentialFeatureSelector
from sklearn.compose import ColumnTransformer, make_column_selector
from sklearn.pipeline import make_pipeline

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import GridSearchCV

from sklearn import set_config
set_config(display='diagram') # shows the pipeline graphically when printed

num_pipeline = Pipeline([
        ('imputer', SimpleImputer(strategy='median')),
        ('scaler', StandardScaler())
    ])
cat_pipeline = Pipeline([
        ('imputer', SimpleImputer(strategy='most_frequent')),
        ('cat_encoder', OneHotEncoder(sparse=False, drop='first')) # returns a
    ])

SimpleImputer.get_feature_names_out = StandardScaler.get_feature_names_out  # pat

prep_pipeline = ColumnTransformer([
    ('num', num_pipeline, make_column_selector(dtype_include=np.number)),
    ('cat', cat_pipeline, make_column_selector(dtype_include=object))
])

full_pipeline = Pipeline([
    ('prep', prep_pipeline),
    ('select', SequentialFeatureSelector(LinearRegression(), n_features_to_select
    ('linear', LinearRegression())
])

full_pipeline
```



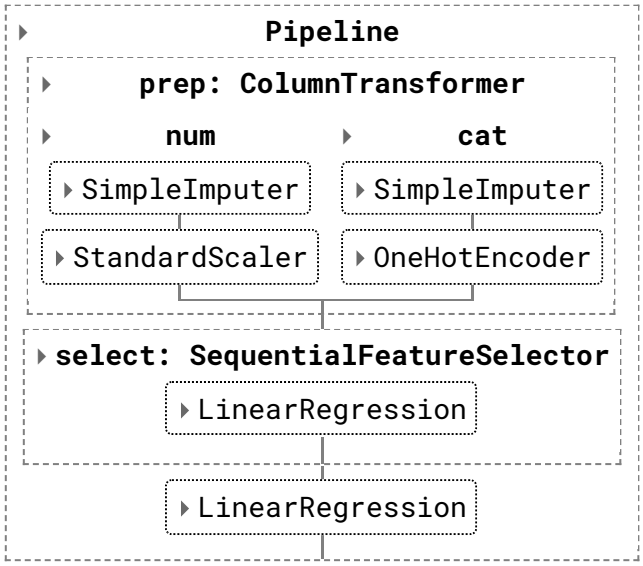
To tune the pipeline, we need the names of the hyper parameters. Let's print them so that we know how to refer to them.

```
full_pipeline.get_params().keys()
```

```
dict_keys(['memory', 'steps', 'verbose', 'prep', 'select', 'linear',
'prep__n_jobs', 'prep__remainder', 'prep__sparse_threshold',
'prep__transformer_weights', 'prep__transformers', 'prep__verbose',
'prep__verbose_feature_names_out', 'prep__num', 'prep__cat',
'prep__num__memory', 'prep__num__steps', 'prep__num__verbose',
'prep__num__imputer', 'prep__num__scaler', 'prep__num__imputer__add_indicator',
'prep__num__imputer__copy', 'prep__num__imputer__fill_value',
'prep__num__imputer__missing_values', 'prep__num__imputer__strategy',
'prep__num__imputer__verbose', 'prep__num__scaler__copy',
'prep__num__scaler__with_mean', 'prep__num__scaler__with_std',
'prep__cat__memory', 'prep__cat__steps', 'prep__cat__verbose',
'prep__cat__imputer', 'prep__cat__cat_encoder',
'prep__cat__imputer__add_indicator', 'prep__cat__imputer__copy',
'prep__cat__imputer__fill_value', 'prep__cat__imputer__missing_values',
'prep__cat__imputer__strategy', 'prep__cat__imputer__verbose',
'prep__cat__cat_encoder__categories', 'prep__cat__cat_encoder__drop',
'prep__cat__cat_encoder__dtype', 'prep__cat__cat_encoder__handle_unknown',
'prep__cat__cat_encoder__sparse', 'select__cv', 'select__direction',
'select__estimator__copy_X', 'select__estimator__fit_intercept',
'select__estimator__n_jobs', 'select__estimator__normalize',
'select__estimator__positive', 'select__estimator',
'select__n_features_to_select', 'select__n_jobs', 'select__scoring',
'linear__copy_X', 'linear__fit_intercept', 'linear__n_jobs',
'linear__normalize', 'linear__positive'])
```

From these, we need to search over the values of `select__n_features_to_select`. We have 17 features. So, we'll select from 1 to 17 features in a grid search.

```
from sklearn.model_selection import GridSearchCV

param_grid = ({
    'select__n_features_to_select': np.arange(1, 17, 1)
    # Now that we know the number of columns there are we are using integer value
    # to explore 1 to 16 columns.
})

grid_search = GridSearchCV(full_pipeline, param_grid, cv=3, scoring='neg_root_mea
grid_search.fit(train_X, train_y)
grid_search.best_estimator_
```



Let's examine the top models.

```
1 cv_res = pd.DataFrame(grid_search.cv_results_)
2 cv_res.sort_values(by="mean_test_score", ascending=False, inplace=True)
3 cv_res.head(5)
```

_time	param_select__n_features_to_select	params	split0_test_sc
01538	11	{'select__n_features_to_select': 11}	-1130.081
00121	14	{'select__n_features_to_select': 14}	-1135.064
00057	13	{'select__n_features_to_select': 13}	-1135.783
02251	12	{'select__n_features_to_select': 12}	-1135.790
00123	10	{'select__n_features_to_select': 10}	-1131.626

We see that using 11 features leads to most accurate model.

Colab paid products - Cancel contracts here

✓ 31s completed at 10:08 AM